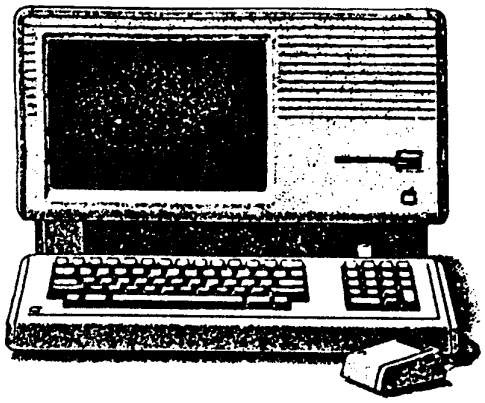
Document# 45

Apple Lisa Information



FILE NAME

Lisa Pascal Review

DISK #

COMMENTS

Apple Doc
Al Hoffman

David T. Craig
736 Edgewater, Wichita, Kansas 67230
(316) 733-0914

Lisa

"45-0.PICT" 167 KB 2000-12-24 dpi: 300h x 300v pix: 1913h x 2911v

Lisa Pascal

Albert A. Hoffman
Apple Computer, Inc.
USUS National Meeting
April 23, 1983

This talk continues that by Barry Haynes,* who spoke on the topics Clascal (a classes extension built into Lisa Pascal) and the Toolkit (a set of Clascal-based units aiding implementation of application programs on Lisa). I will speak on the Lisa Pascal development environment, named the WorkShop; on the differences between Lisa Pascal and UCSD Pascal; and on the implementation issues of putting a UCSD Pascal on Lisa.

A. The WorkShop

The development environment on Lisa is named the WorkShop. It should be immediately familiar to users of UCSD Pascal, as it is promptline-based, with a hierarchical set of command lines which are displayed at the top of each screen. The WorkShop is not UCSD Pascal, however; it simply has strong influences from UCSD Pascal. The command lines are a little different, for example systemwide environmental modifications are placed on a separate promptline called the System Manager.

The Editor is completely different from the UCSD Pascal Editor. It is a Lisa-oriented, mouse-based Editor that:

- can read multiple files, on-screen, at once. Each new file occupies a separate window on the screen.
- The Editor is a "resident process". The user can completely leave the Editor, perform compilation, filing or other functions, and return to the suspended editing environment. Each file being edited is returned to the exact state on the screen it was in before the editor was suspended. Suspension and return are fast (less than one second).
- The Editor has multiple fonts, including very small fonts that are useful to get more lines on the screen.
- The Editor features cutting and pasting between files, using the mouse.
- The Editor permits very large, but memory-based, files (200 or more blocks).
- The Editor stores text in UCSD-format textfiles.

The Pascal compiler is a native-code (68000) compiler. In order to compile and run a program, one must go through three separate steps: compilation, code generation, and linking. Linking is required for interfacing to the standard run-time library and to units and assembly language.

The system supports a rather sophisticated, parameterizable exec file language. This language supports nested IF conditions, recursion, and interactive user input. The threefold task of compiling a Pascal program is eased by the use of an exec file. Exec files are "killed" when a compilation or system error is detected.

Intrinsic libraries exist in the system, but are not user-modifiable in the first release. Regular units are available.

The system supports background and resident processes. There are WorkShop prompts to make a program resident (stays in virtual memory between executions) and to run a program in the background. Background execution is not fully developed yet; screen I/O is not separated from the foreground process. It is, therefore, most useful for a program with a separate output channel (text spooling/printing, plotting, etc.).

The system is equipped with a low-level debugger. The debugger may be invoked to run a program, or is invoked automatically when an execution error occurs. The debugger knows about procedure names, but not variable names or Pascal source statements. Interaction with the debugger requires a little familiarity with 68000 assembly language; however, the appearance of procedure names in offsets and jumps serve as useful guideposts in the code. The debugger supports breakpointing, tracing, hex and decimal calculation, disassembly, memory dumping, and other functions.

Various utility programs are a part of the WorkShop, including Diff, a program to list differences in texts; Find, a program to locate a string in a wildcard list of textfiles; and several program crossreference utilities. A program called Transfer permits terminal-style, RS-232 interaction with other computers and is especially useful for downloading program sources from other computers.

The capability for copy-protecting software is built in to the WorkShop. Software which has been copy-protected with this mechanism will run on any Lisa, but its copies will only run on one machine (the first machine on which a copy has been made).

B. Lisa Pascal Language

The Lisa Pascal Language is based on UCSD Pascal and ISO Pascal. There are some modifications to support system development, most notably the Clascal extension to support classes. Other extensions include the "pointer" and "@" operators, which create pointer types; type "longint" which is 32-bit integers; hex constants (ie. $3FE); procedure and function parameters with type-checked parameter lists. Lisa Pascal also supports very large procedures, and allows for more arbitrary segmentation than UCSD Pascal.

*Headed Workshop development.

"45-1.PICT" 303 KB 2000-12-24 dpi: 300h x 300v pix: 2150h x 3089v

The major differences in Lisa Pascal, affecting a port from UCSD Pascal, are the following:

- stronger type-checking in Lisa Pascal, which has name-compatibility a la ISO Pascal rather than structure compatibility. Variables are of compatible type if they are declared with the *same type identifier.*
- All standard, supplied units are different from those supplied with Apple Pascal or Version IV Pascal. There is no chaining nor decimal-based long integers.
- Unit I/O and Volume I/O are not available in Lisa Pascal. All I/O operations must be done through the file system.
- Extended comparisons of complex types (except packed array of char) is not implemented.
- Moveleft, Moveright, and Scan are restricted in the types of parameters they accept. However, a for-loop in Lisa Pascal is virtually as fast.
- For-loop control variables must be local, another ISO influence.
- IOResult values are different and more numerous.
- Filenames are a little different from UCSD Pascal; they are not hierarchical, but up to 32 characters are allowed in volume and file names and the character "-" is used as a delimiter.
- The assembly language interface is currently more restricted; .PUBLIC's and .PRIVATE's are not permitted.
- Automatic file closing is done at process exit rather than procedure exit.
- The runtime stack structure is extremely different. There is more than 500K bytes of stack/heap space available for program data.
- Native code is used; many algorithms will obtain performance using standard Pascal constructs where special tricks had to be used in UCSD Pascal.

There are some aspects of programming Lisa-like applications that will affect a non-trivial portation of programs. Lisa has a *graphics orientation* in its user interface that presents a higher-bandwidth channel to the user's mind. Programs have to accommodate this channel by providing more detail about the information presented to the user. (For example, if the output is an error message it should go in an *alert box.*) The Toolkit provides mechanisms to implement this higher-bandwidth output.

Lisa programs have a *mouse-menu orientation* that is different from many UCSD Pascal programs, where there is a trend to use a command-line orientation.

Lisa programming involves an *event-driven orientation* that could affect the overall structure of programs. Programs must be prepared to accept events such as mouse clickings and screen updates at essentially any time. Again, the Toolkit provides mechanisms for implementing the event-driven orientation.

Last, but perhaps most important, Lisa programs are expected to have a *modeless orientation.* This means that the user can't find himself or herself stuck in a command sequence or program mode, unable to access other useful features immediately. Many UCSD Pascal programs are highly moded, including the UCSD system itself. It is very difficult to get the modes out of a developed program; but the trend in the 80's will be to develop very integrated packages in which modes do not exist.

C. What about UCSD Pascal on Lisa?

What is the best way to integrate UCSD Pascal into a foreign file system and operating system environment? Two common and useful ways of supporting UCSD Pascal are to have UCSD Pascal "take over" the machine (replacing the host operating system), and to have an "embedded UCSD Pascal system" occupying the same position as an arbitrary program within the host operating system.

A "taking over" of the machine is common; in fact, Lisa will run the ZENIX* operating system in just such a mode. In this scenario, one expects to find programmers and users falling into independent groups, groups who are associated by using the same hardware, but who really have little else in common. These groups tend to use independent sets of software, and the software and data are not shareable among the groups. This is a common situation on the Apple II, the IBM PC, and on mainframes and minis. One disadvantage of this is that hardware manufacturers, especially third party peripheral manufacturers, are forced to provide firmware, manuals, and sometimes hardware modifications for each software system supported. Another problem faced by end-users is the problem of sharing hard-disk media among disjoint systems. The file system of most operating systems likes to "take over" the entire media. Owners of hard disks then have to decide which system they are going to be loyal to, unless they can afford to buy multiple hard-disk systems. Networking efforts, also, typically have to be duplicated in each system. Worst of all, each separate system is a separate MODE and rebooting is required to switch to a different system (mode).

Overcoming these problems, what would it be like to have a UCSD Pascal system "taking over" the Lisa? Would it be worth it? Since the various host operating systems on the machine compete for manufacturer, etc, resources, we can afford to have only a few operating systems supported. For the UCSD system to be competitive with other operating systems, it will have to provide advantages to a large body of Lisa users. As we all know, the primary benefit of UCSD Pascal is its portability. The system allows developers and the common user community to write, sell and trade programs that work on the IBM PC, the Apple II, the Sage, and so on. However, I would like to quote Dick Sites, my former professor at UCSD, who asserted that "nothing scales up by a factor of ten." The Apple II is 64K, while Lisa is well over 640K in memory space. IBM and Sage are somewhere in between. To which size system is UCSD Pascal best suited? Currently, it can use up to about 128K of RAM before it must resort to less effective measures such as virtual disk storage. Within this space, both II.0 and IV.1 have a limit of 64K for data space. This is already hurting the progress of the system on the IBM, and would be quite limiting on Lisa. So there we have a need for an expansion to UCSD, perhaps an incompatible expansion, if we want it to be competitive with other operating systems.

* Xenix ?

"45-2.PICT" 372 KB 2000-12-24 dpi: 300h x 300v pix: 2191h x 3046v

UCSD Pascal is interpreted; or at best, partially interpreted. Is this an advantage or a disadvantage on Lisa? Native code running on Lisa is about ten times faster than interpreted code. However, the size of native code from the WorkShop compiler is about 1.3 times larger. It will probably take some skillful marketing to make this seem like an advantage for UCSD Pascal.

The Lisa Pascal language is a little more powerful than UCSD Pascal's language. Details of this have been discussed above.

Disk storage on Lisa consists of two 871Kb disk drives and a 5Mb hard disk. Future directions indicate much larger storage sizes. UCSD Pascal would have to manage this amount of storage efficiently to be competitive. With its continguous allocation, directory limitations, and lack of subdirectories, it's probably safe to say that it suffers in comparison to the UNIX operating environment in this area.

Currently, the UCSD Pascal system's concurrency features are not as well developed as those of either ZENIX* or the native Lisa operating system.

The conclusion is that, as a separate, non-hosted operating system, UCSD Pascal in its current form would not be very competitive with existing Lisa systems, including the host operating system and the ZENIX* system.

The other mechanism to consider is to host the UCSD system within the host operating system. On Lisa, this would consist of encapsulating UCSD Pascal in a "vanilla window" on the screen, and embedding a UCSD file system within a host system file on the larger media. (And on the floppy disks, since the Lisa OS doesn't support Unit I/O to floppies).** Such an arrangement, while not without some of the drawbacks of the previously described system, would permit portable UCSD programs to operate on the Lisa machine, and, with UCSD being one of many windows available concurrently, a user could easily switch between UCSD Pascal programs and other Lisa applications, such as the LisaWrite or the Calculator. Apple's prior experience indicates that end-users strongly desire better integration with other (native) programs while using UCSD Pascal, so this is an advantage. Our experience also shows, though, that software developers eventually want better access to all features provided in the host environment when running UCSD Pascal. The better (more successful) applications will probably be those that are fully integrated into the Lisa programming environment. The conclusion is that the chief advantages of UCSD Pascal, being portability and an existing basis of users and programs, can be as well supported when UCSD Pascal is embedded in a "vanilla window" in the host OS as they would be in a "machine take over" mode. An embedded UCSD Pascal system would be useful for getting up portable programs quickly, but not for long-term applications programming, or more sophisticated programming.

US

* Xenix?

** Not true; private calls exist for direct unit I/O, but only strictly for OS internal use and not documented for public releases.

"45-3.PICT" 197 KB 2000-12-24 dpi: 300h x 300v pix: 2167h x 1854v